Universitat Autònoma de Barcelona

Dipòsit digital de documents de la UAB

This is the **published version** of the article:

Pérez González, Valentín; Quirós Jiménez, José; Juan Francés, Robert. Realización de una aplicación web para la consulta de los datos de los trabajos realizados por los servicios de apoyo al desarrollo en Santa Coloma de Gramenet. 2013. 44 p.

This version is available at https://ddd.uab.cat/record/181500

under the terms of the BY-NC-ND license

Realización de una aplicación web para la consulta de los datos de los trabajos realizados por los servicios de apoyo al desarrollo en Santa Coloma de Gramenet.

Autor: Valentín Pérez González

Tutores: José Quirós Jiménez
Robert Juan Francés

Màster en Tecnologies de la Informació Geogràfica, 14a Edició
Departament de Geografia
Universitat Autònoma de Barcelona
Entidad colaboradora: Projectes i Gestió de Serveis Socials, S. L. (PROGESS, S.L.)

PROGESS

UAB Universitat Autònoma de Barcelona
Departament de Geografia

14mtig 2012

Agradecimientos

Quiero agradecer a la empresa PROGESS y en especial a Robert Juan por darme esta oportunidad para crecer como desarrollador de Sistemas de Información Geográfica.

También agradecer a todo el personal del Laboratori d'Informació Geogràfica i de Teledetecció (LIGIT) y a los profesores de la 14ª edición del Màster en Tecnologies de la Informació Geogràfica, especialmente a José Quirós por guiarme durante el proyecto final.

Y por último agradecer a todos mis compañeros del máster por estar siempre dispuestos a echar una mano para resolver cualquier duda.

Resumen

El presente documento es la explicación del proyecto final del Màster en Tecnologies de la Informació Geogràfica, 14ª. Edición. El proyecto es fruto del convenio de colaboración entre la Universitat Autònoma de Barcelona y la empresa Projectes i Gestió de Serveis Socials, S.L.

La finalidad del proyecto es la creación de un visor cartográfico web, aunque para ello se debe crear una aplicación que transfiera datos automáticamente desde una base de datos de Acces a MySQL mediante Visual Basic 2010 Express. Para realizar el visor cartográfico web se ha utilizado MapServer como servidor de mapas, el lenguaje de programación Javascript y las librerías OpenLayers, ExtJS y GeoExt. Por último se plantean una serie de posibles mejoras del visor, tanto a nivel de tecnologías utilizadas como de funcionalidades.

Palabras clave

MySQL, MapServer, OpenLayers, Sistemas de Información Geográfica, Visual Basic 2010 Express, Javascript

Resum

El present document és l'explicació del projecte final del Màster en Tecnologies de la Informació Geogràfica, 14ª. Edició. El projecte neix del conveni de col·laboració entre la Universitat Autònoma de Barcelona i l'empresa Projectes i Gestió de Serveis Socials, S.L.

La finalitat d'aquest projecte és la creació d'un visor cartogràfic web, tot i que per fer-ho s'ha de crear una aplicació que transfereix dades automàticament des d'una base de dades de Microsoft Acces a MySQL mitjançant Visual Basic 2010 Express. Per realitzar el visor cartogràfic web s'ha utilitzat MapServer com servidor de mapes, el llenguatge de programació Javascript i les llibreries OpenLayers, ExtJS i GeoExt. Per últim es plantegen una sèrie de possibles millores del visor, tant a nivell de tecnologies utilitzades com de funcionalitats.

Paraules clau

MySQL, MapServer, OpenLayers, Sistemes d'Informació Geogràfica, Visual Basic 2010 Express, Javascript

Índice

Índice de figuras

Aplicación de transferencia de datos

Visor web

Programación del visor

Resultados

1. Introducción

El presente documento expone el proyecto final del Màster en Tecnologies de la Informació Geogràfica, 14ª. edición, organizado por el Departament de Geografia de la Universitat Autònoma de Barcelona.

Este proyecto es fruto de la colaboración entre la UAB y la empresa PROGESS (Projectes i Gestio de Serveis Socials), siendo el tercer proyecto realizado entre ambas instituciones. El primero de ellos se llamaba *Sistema d'Informació per a la gestió de la intervenció social en medi obert* y fue realizado por Robert Juan Francés en la 10ª edición del Màster en Tecnologies de la Informació Geogràfica y el objetivo era gestionar la información de un servició de intervención en medio abierto para jóvenes. El segundo proyecto fue el llamado *Sistema de Información Geogràfica para visualuzación y consulta de datos del servicio de inserción social del Ayuntamiento de Barcelona*, realizado por Karla Abad Sacoto en la 11ª edición del MTIG.

La empresa PROGESS, S.L. ofrece servicios sociales de carácter público en diferentes campos de actuación como pueden ser la infancia, inmigración, personas excluidas socialmente o la acción social comunitaria. En este último campo es donde se podría incluir este proyecto, ya que se trata de realizar un visor cartográfico web de las acciones llevadas a cabo por el servicio para gestionar demandas e informaciones vecinales que hacen referencia a situaciones que puedan convertirse en potencialmente problemáticas para la convivencia.

1.1 Estructura del proyecto

El proyecto está divido en dos partes. La primera parte sería la migración de los datos de la base de datos de Acces a la de MySQL y la aplicación de Visual Studio 2010 para automatizar el paso de información de Acces a MySQL. Mientras que la segunda parte es la creación del visor cartográfico con MapServer, OpenLayers, ExtJS y GeoExt.

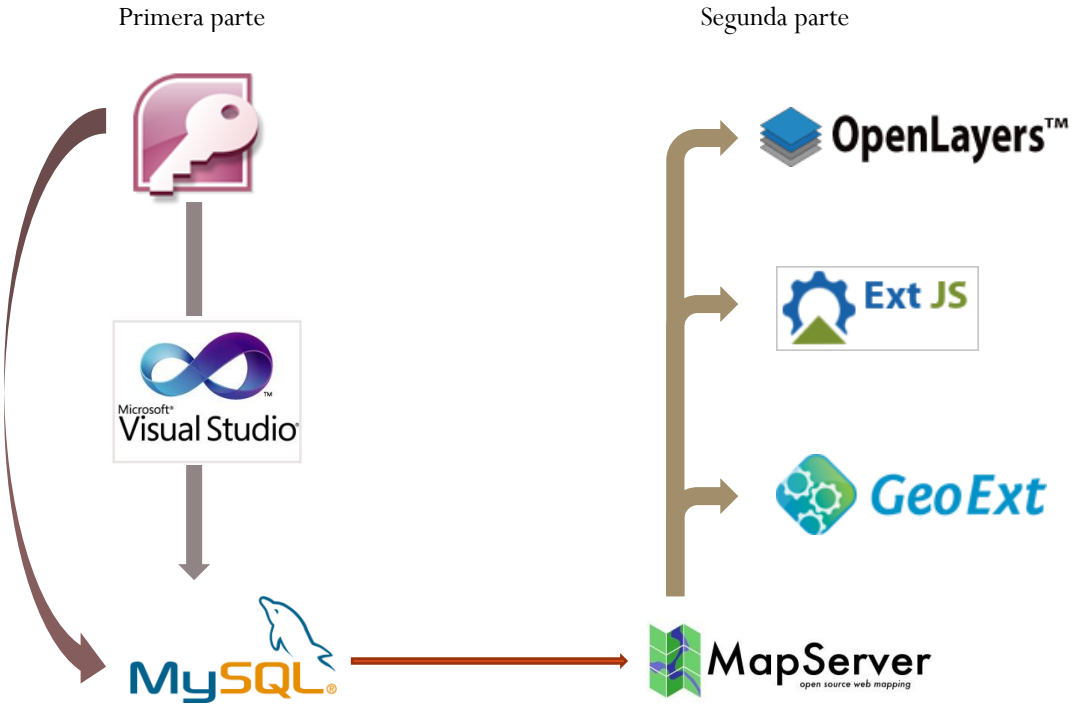


Figura 1: Esquema del proyecto

En la figura 1 se muestra el planteamiento del proyecto con las dos partes diferenciadas. La primera parte consta de la importación de la base de datos de Acces a MySQL y la elaboración de la aplicación de actualización de datos en Visual Basic.NET. La segunda parte consiste en la realización del visor cartográfico mediante MapServer con los datos de MySQL y las librerías de OpenLayers, Ext y GeoExt.

	17/9	24/9	1/10	8/10	15/10	22/10	29/10	5/11	12/11	19/11	26/11	3/12	10/12
Planteamiento del proyecto													
Desarrollo de la aplicación													
Diseño de la página web i de la representación cartográfica													
Desarrollo de la aplicación web													

Figura 2: Calendario del proyecto

2. Objetivos

2.1 Objetivos generales

El principal objetivo del proyecto final es crear un visor cartográfico web para facilitar el trabajo de los técnicos que tienen que trabajar con las comunidades de vecinos. Este visor se hará a partir de los datos que ellos mismos introducen a una base de datos de Microsoft Acces. Para hacerlo es necesario realizar las siguientes tareas:

- Crear una Base de datos de MySQL
- Realizar una aplicación que actualice la base de datos de MySQL desde la de Microsoft Acces.
- Creación del visor.

2.2 Objetivos específicos

- Creación de la base de datos de MySQL con la misma estructura de la base datos ya existente de Microsoft Acces para facilitar la actualización de la información.
- Realizar una aplicación con Visual Studio 2010 que se ejecute una vez al día per pasar la información de las tablas de Acces a MySQL. Para ejecutar la se utilizará el programador de tareas de Windows XP.
- El visor web es realizará con la información de las actuaciones en comunidades de vecinos con diferentes tipos de simbolización según las diferentes categorías. También incluirá una tabla para ver los diferentes atributos de cada actuación.

3. Aplicación de transferencia de datos desde Microsoft Acces a MySQL

3.1 Análisis de requerimientos

- Pasar datos de Microsoft Acces a MySQL de forma automática.
- No sólo es añadir datos, también se tienen que modificar los datos existentes si han cambiado en la base de datos de Acces.
- Que el proceso de actualización no perjudique el rendimiento del servidor.
- El estudiante puede elegir con que tecnología y lenguaje de programación realiza la aplicación.

3.2 Solución metodológica

La tecnología elegida para realizar la aplicación es Visual Studio 2010 Express, ya que es una versión gratuita y que permite realizar la conexión con las bases de datos de Microsoft Acces y MySQL, y trabajar con sus datos. Además el estudiante ya está familiarizado con esta tecnología. Las bases de datos de Acces y MySQL están impuestas por la empresa.

3.2.1 Tecnologías utilizadas

Microsoft Acces: Es un programa de gestión de bases de datos relacionales creado por la compañía Microsoft que está incluido dentro del paquete Microsoft Office.

MySQL: Es un sistema de gestión de bases de datos relacional, multihilo y multiusuario perteneciente a Oracle Company desde el año 2009. Tiene licencia GNU GPL para cualquier usuario que quiera utilizarlo bajo los términos de esta licencia, en caso contrario deben comprar una licencia comercial.

phpMyAdmin: Software gratuito con licencia GNU GPL escrito en php creado para la administración de MySQL mediante el uso de un navegador web. Este administrador permite crear y borrar bases de datos, tablas, registros, ejecutar cualquier sentencia SQL, dar y quitar permisos, importar y exportar información, etc.

Microsoft Visual Basic 2010 Express: Versión gratuita del software Visual Studio 2010 creado por Microsoft. Sirve para crear aplicaciones web o para entornos de Microsoft.

3.3 Estructura e importación de la base de datos

La base de datos que se ha de utilizar para para crear el visor está en formato Acces, por lo que deberá ser importada a MySQL. Como se puede apreciar en la figura 3 la base de datos de origen contiene 6 tablas relacionadas entre sí. Las tablas realmente interesantes, que son las que deberemos actualizar periódicamente, son tblccvv que contiene las direcciones y coordenadas de las comunidades de vecinos y tblaccio con la información relevante de cada actuación.

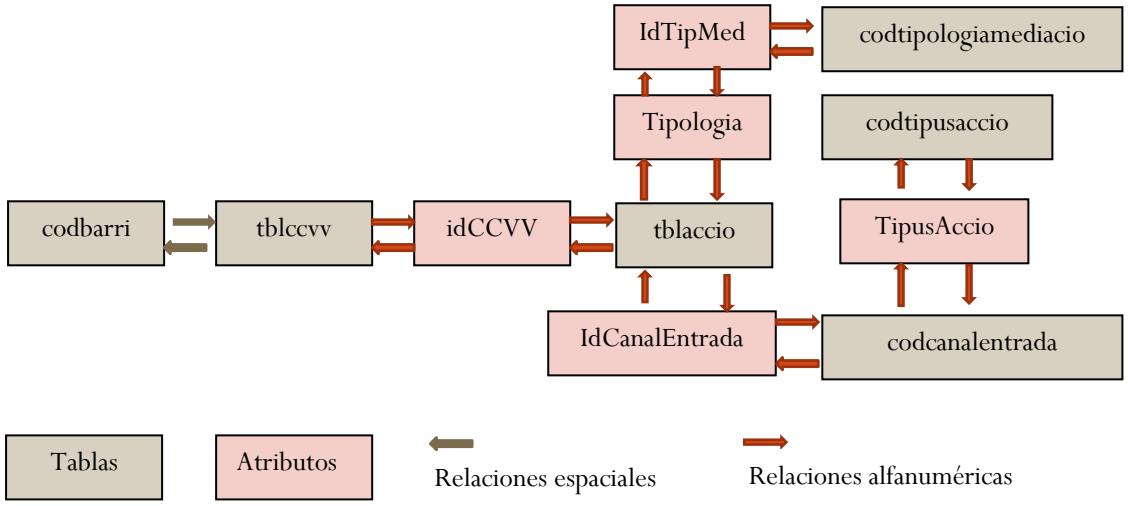


Figura 3

Antes de realizar la aplicación se deberá crear la base de datos de MySQL importando las tablas de Acces con el asistente para exportación de Microsoft Acces, eligiendo el driver de MySQL OBDC.

Figura 4

Figura 5

Figura 6

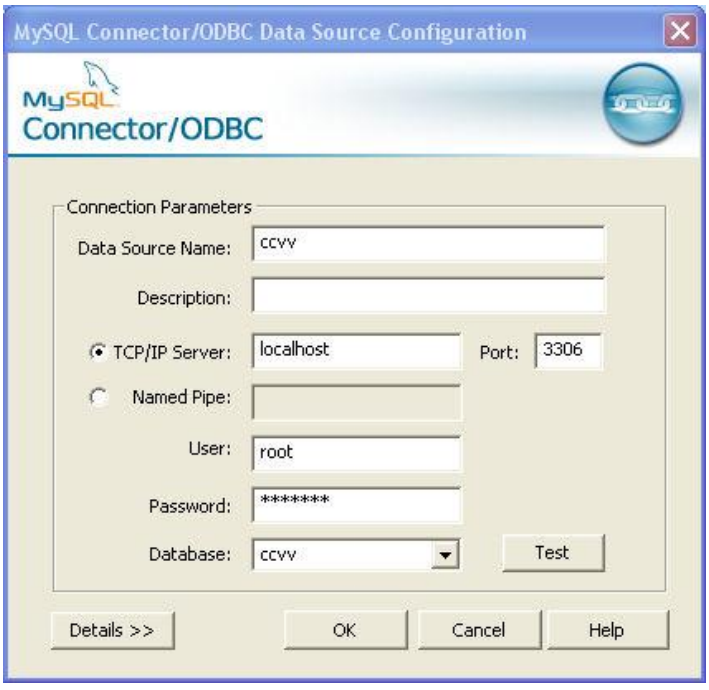


Figura 7

Con la tabla tblaccio hay problemas de compatibilidad, ya que contiene campos de Verdadero / Falso que en MySQL se representan con 1 y 0, por lo que primero tendremos que convertir esta tabla a formato xls y posteriormente importarla desde MySQL, aunque dichos campos aparecen como string con el texto verdadero o falso, por lo que, para optimizar recursos es recomendable convertir los verdaderos en 1 y los falsos en 0 y elegir como tipo de campo tinyint, de esta forma cuando la aplicación de actualización funcione no dará problemas.

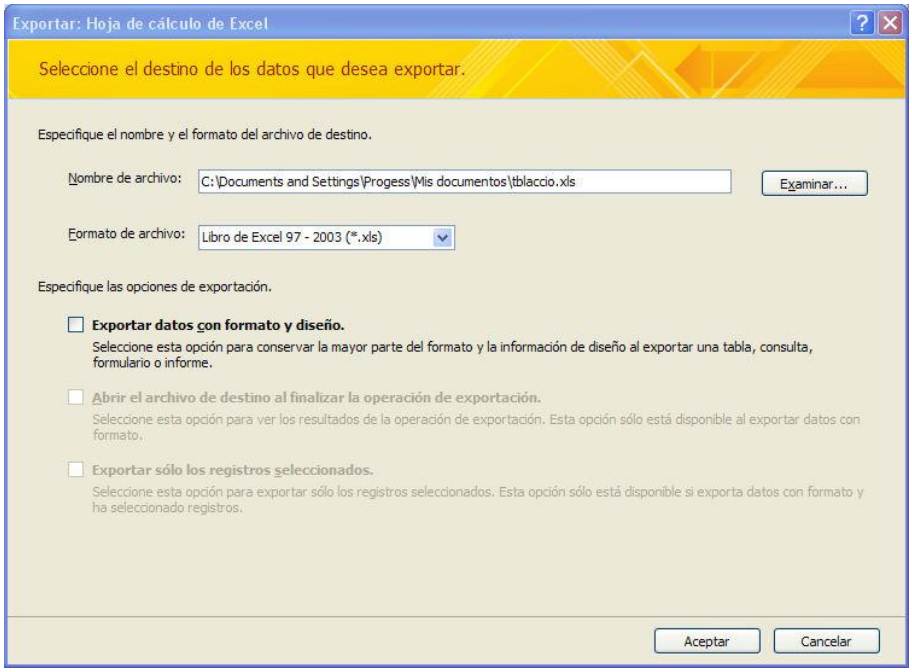


Figura 8

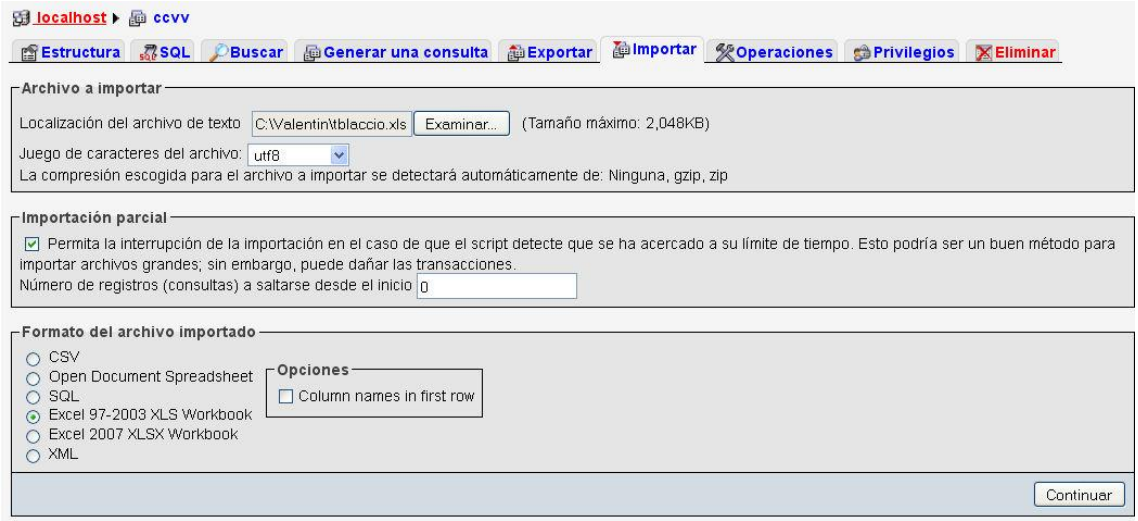


Figura 9

3.4 Diseño funcional

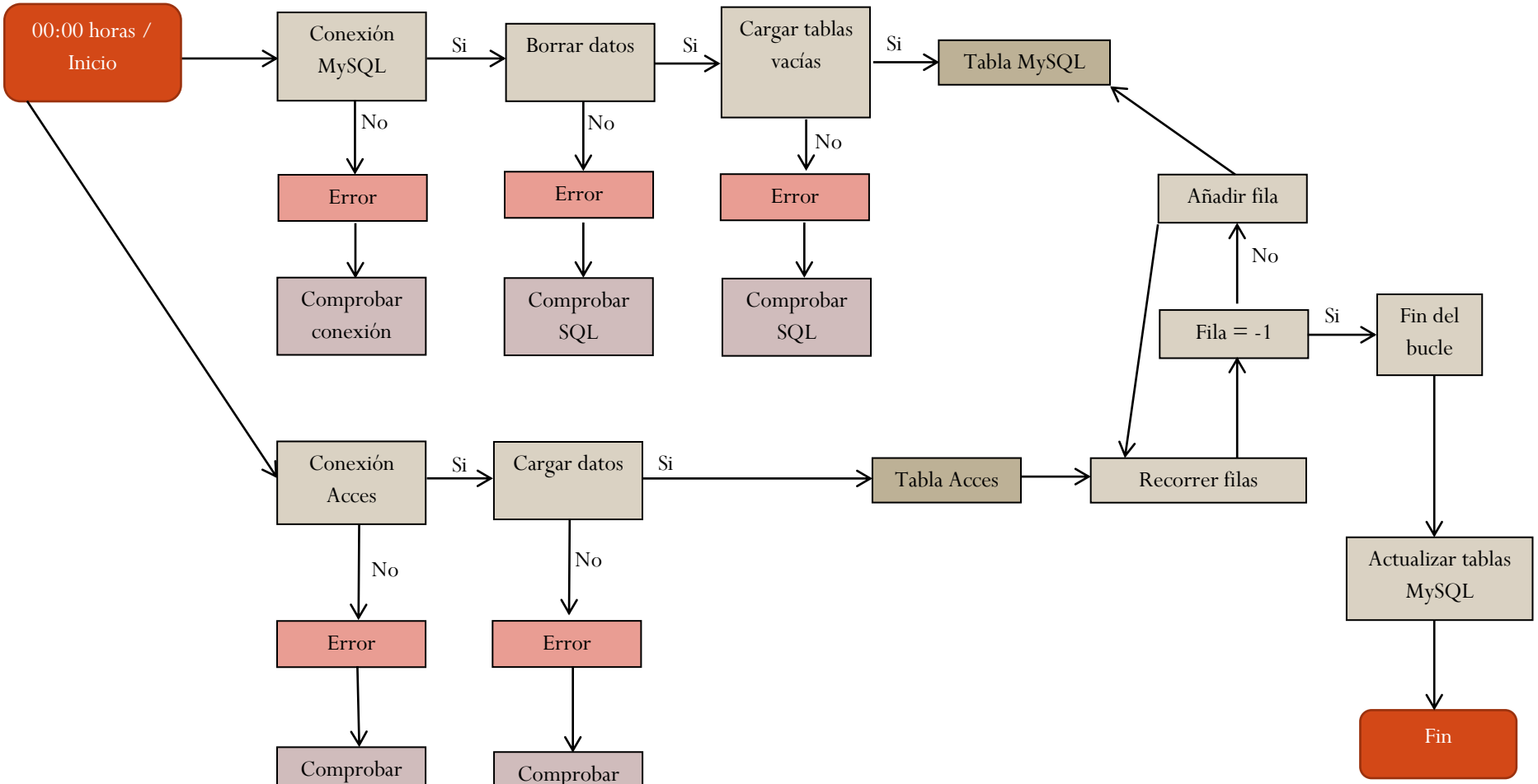


Figura 10

3.5 Programación de la aplicación

Una vez tenemos las dos bases de datos el primer paso para hacer la aplicación es importar los conectores de les bases de datos. La base datos de Acces no tiene ninguna complicación, pero para conectar con MySQL, se debe instalar previamente el conector adecuado y añadir la referencia correspondiente al proyecto.

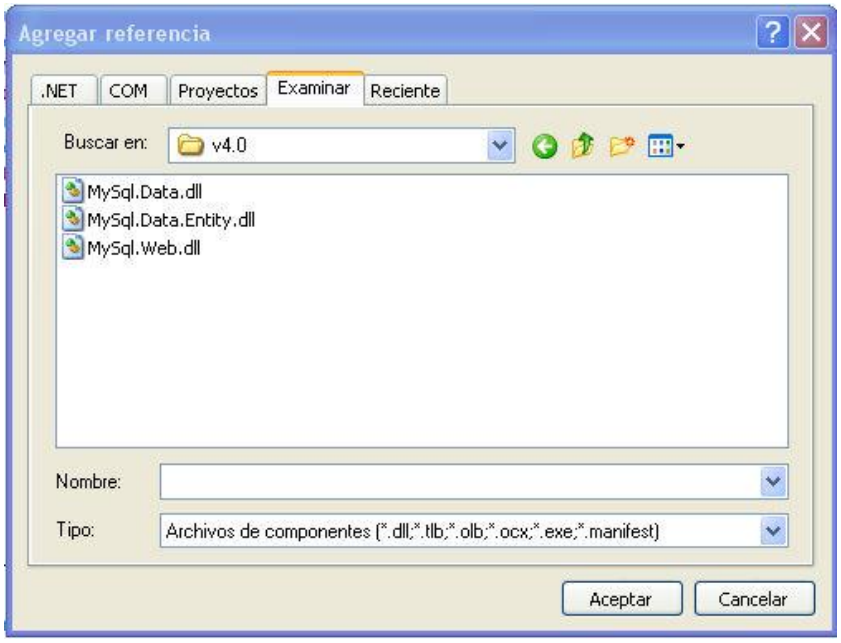


Figura 11

Ahora si se podrá importar el conector de MySQL.

Conectores de Acces

```
Imports System.Data.OleDb
Imports System.Data
```

Conectores de MySQL

```
Imports MySql.Data
Imports MySql.Data.MySqlClient
```

El segundo paso es crear las variables para conectarse con ambas bases de datos.

```
Const urlAcces = "C:\Valentin\datos_origen\EII_be.accdb"
Dim aConn As New OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" & urlAcces & ";User Id=;Password=;")
Dim mConn As New MySqlConnection("Data Source=localhost;Database=ccvv;User ID=root;Password=****;")
```

La primera variable es la localización del fichero de la base de datos de Acces. La segunda es la conexión a la base de datos de Acces y la tercera la conexión a la base de datos de MySQL.

```
Dim ds As New DataSet
Dim aSqlCcvv As String = "SELECT * FROM tblccvv"
Dim aSqlAccio As String = "SELECT * FROM tblaccio"
Dim daAccesCcvv As New OleDbDataAdapter(aSqlCcvv, aConn)
Dim daAccesAccio As New OleDbDataAdapter(aSqlAccio, aConn)
Dim mSqlCcvv As String = "SELECT * FROM tblccvv"
Dim mSqlAccio As String = "SELECT * FROM tblaccio"
Dim daMysqlCcvv As New MySqlDataAdapter(mSqlCcvv, mConn)
Dim daMysqlAccio As New MySqlDataAdapter(mSqlAccio, mConn)
Dim borrarCCVV As New MySqlCommand("TRUNCATE TABLE tblccvv", mConn)
Dim borrarAccio As New MySqlCommand("TRUNCATE TABLE tblaccio", mConn)
```

En este código creamos el dataset, los data adapter y los query necesarios para trabajar con las tablas de las dos bases de datos. En primer lugar se crea el dataset para almacenar los datos de las tablas, que será llamado ds, después las sentencias SQL para seleccionar los datos que nos interesan de la base de datos, que en este caso son dos tablas tblaccio y tblccvv. Posteriormente los Data Adapter de Acces indicando la variable con la sentencia SQL y la conexión a la base de datos. Repetimos el proceso para la base de datos de MySQL, pero en este caso añadimos las variables borrarCCVV y borrarAccio como MySQLCommand y en ellas la sentencia truncate table que elimina todos los registros de una tabla pero mantiene la estructura de la misma.

Una vez han sido creadas las variables se ejecutan.

```
aConn.Open()
daAccesCcvv.Fill(ds, "tblccvv")
daAccesAccio.Fill(ds, "tblaccio")
aConn.Close()
mConn.Open()
borrarCCVV.ExecuteNonQuery()
borrarAccio.ExecuteNonQuery()
daMysqlCcvv.Fill(ds, "tblccvvM")
daMysqlAccio.Fill(ds, "tblaccioM")
mConn.Close()
```

Se comienza con la base de datos de Acces, se abre la conexión, se meten las tablas en el dataset y se vuelve a cerrar la conexión. En MySQL se realiza el mismo proceso pero antes de meter las tablas se ejecutan los MySqlCommand de borrado de tablas, así las tablas ya se introducen en el dataset vacías, por último se cierra la conexión.

```
For i = 0 To ds.Tables("tblccvv").Rows.Count - 1
        ds.Tables("tblccvvM").Rows.Add(ds.Tables("tblccvv").Rows(i).ItemArray)

Next

For i = 0 To ds.Tables("tblaccio").Rows.Count - 1
        ds.Tables("tblaccioM").Rows.Add(ds.Tables("tblaccio").Rows(i).ItemArray)

Next
```

Se realiza un bucle para cada tabla para añadir todas las filas de las tablas de Acces en las tablas de MySQL importadas.

```
Dim commandBuilderCC As New MySqlCommandBuilder(daMysqlCcvv)
Dim commandBuilderAC As New MySqlCommandBuilder(daMysqlAccio)
```

Se crean los commandBuilder de los Data Adapter de las dos tablas de MySQL para poder actualizarlas.

```
mConn.Open()
daMysqlCcvv.Update(ds, "tblccvvM")
daMysqlAccio.Update(ds, "tblaccioM")
mConn.Close()
Close()
```

Se abre la conexión de MySQL de nuevo, se actualizan ambas tablas con los datos nuevos, se cierra la conexión y el programa.

El último paso es crear una tarea programada de Windows para que la aplicación se ejecute una vez al día a las 00:00 para no afectar al rendimiento del servidor. Desde Tareas Programadas se selecciona agregar tarea programada y se añade el archivo .exe que Visual Studio ha generado en la carpeta Debug del proyecto (figura 12).

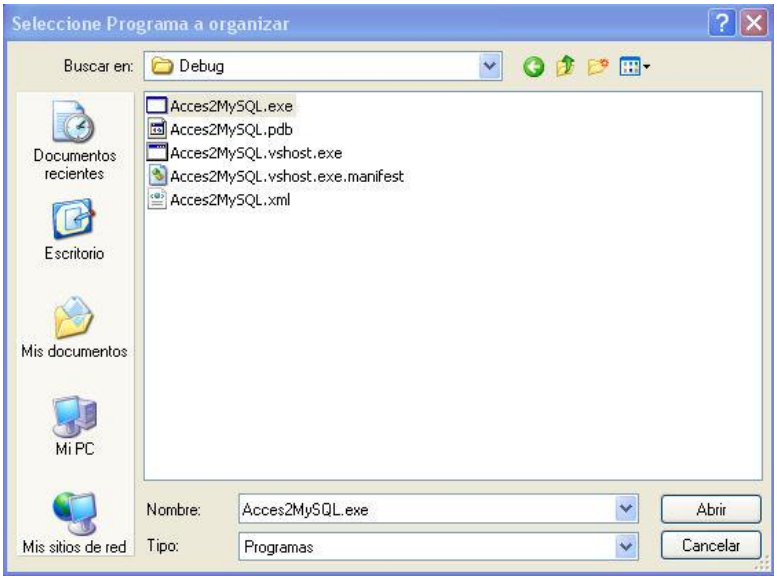


Figura 12

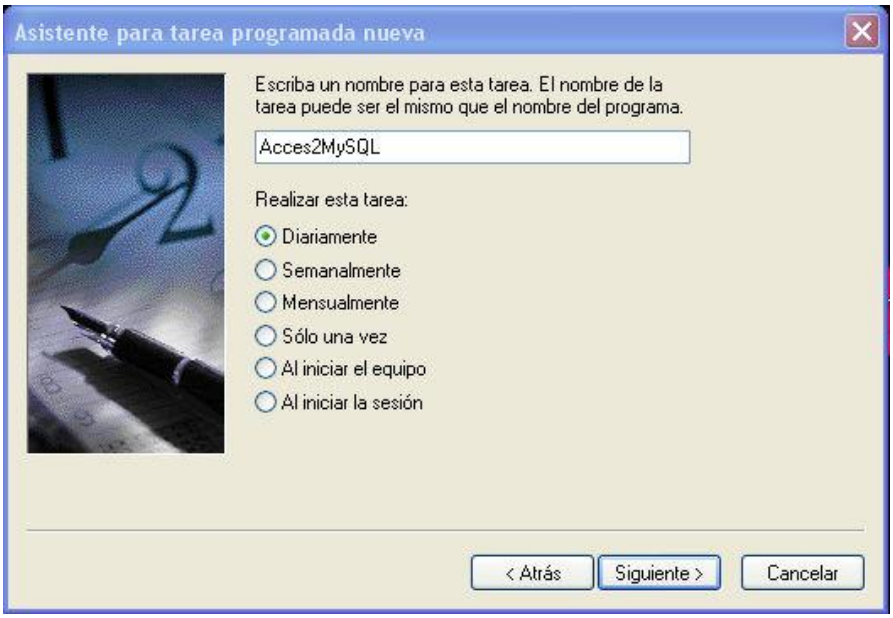


Figura 13

Figura 14

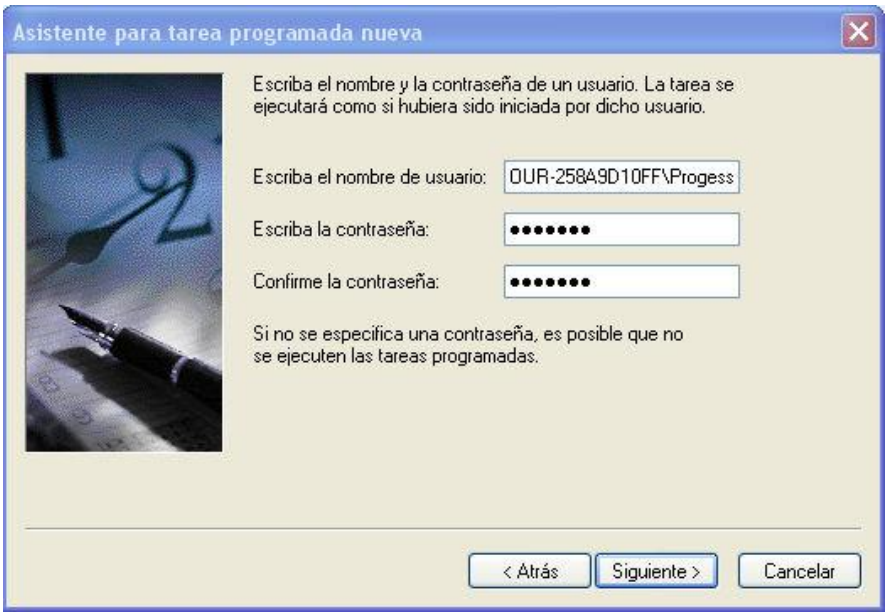


Figura 15

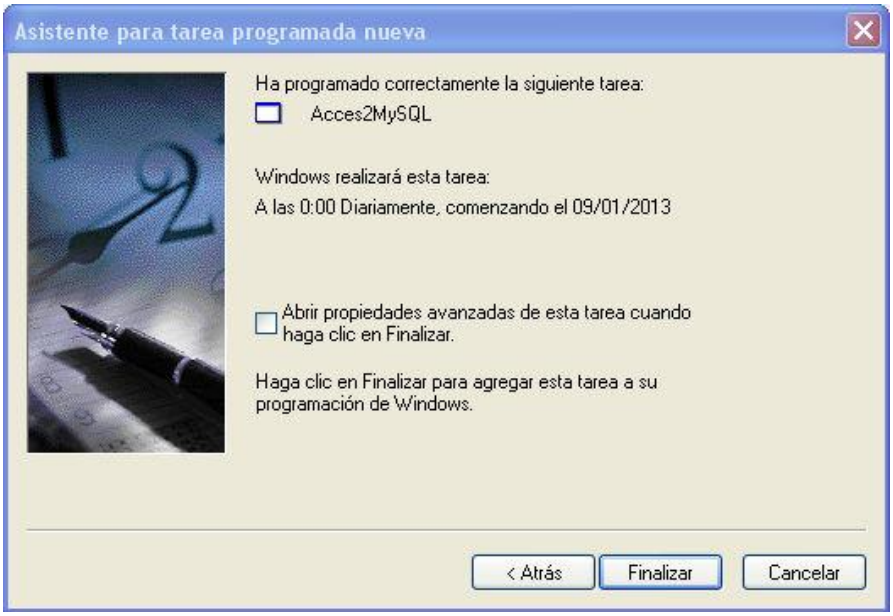


Figura 16

Con el asistente se selecciona la frecuencia de ejecución (figura 13), la hora y fecha de inicio (figura 14) y, por último, se introduce el usuario y la contraseña del usuario que ejecuta la aplicación (figura 15).

4. Visualizador web

4.1. Análisis de requerimientos

- Visualizar datos y la localización de las comunidades de vecinos de Santa Coloma de Gramenet con conflictos.
- La interfaz debe ser intuitiva.
- El estudiante puede elegir la tecnología para realizar el visor, excepto la base de datos de origen que será MySQL.

4.2. Solución metodológica

La principal dificultad de la aplicación es crear la capa de puntos de las comunidades de vecinos a partir de dos campos numéricos de la base de datos que indican la latitud y la longitud de cada punto. Para solventar esta dificultad se ha hecho uso del driver OGR Virtual Format que transforma en capas espaciales tablas con información geográfica en los atributos.

Se utilizará MapServer sobre Apache como servidor de mapas, ya que se encuentra instalado en el ordenador de la empresa se encuentra instalado ms4w que incluye el servidor Apache en su versión 2.2.17 y MapServer en la versión 5.6.5. También se utilizará QuantumGIS para crear los archivos .map propios de MapServer. En cuanto al visor se utilizarán los lenguajes html y javascript con las librerías OpenLayers 2.12, Ext JS 3.4 y GeoExt 1.1 con las que el alumno se encuentra familiarizado. Las pruebas serán realizadas con Firefox 17 y la extensión Firebug 1.11.1. Toda la tecnología utilizada es de código abierto.

4.2.1. Explicación de tecnologías utilizadas

Apache 2.2.17: Servidor HTTP de código abierto lanzado por primera vez en 1995. Es altamente configurable pero no tiene una interfaz gráfica para ello.

Minnesota MapServer 5.6.5: Entorno de desarrollo Web SIG de código abierto desarrollado por la Universidad de Minnesota. Permite publicar datos espaciales y aplicaciones de mapas interactivas, soportando un gran número de formatos tanto de entrada como de salida. En su forma más básica MapServer es un programa CGI, Common Gateway Interface o Interfaz de entrada común, que permite a un navegador web solicitar datos de un programa ejecutado en un servidor web, en este caso el

programa es MapServer y el servidor web es Apache. Cuando se envía una petición a MapServer este usa la información de la URL y el Mapfile para crear una imagen del mapa pedido. Incluso es capaz de devolver la leyenda, escala, mapas de referencia, etc.

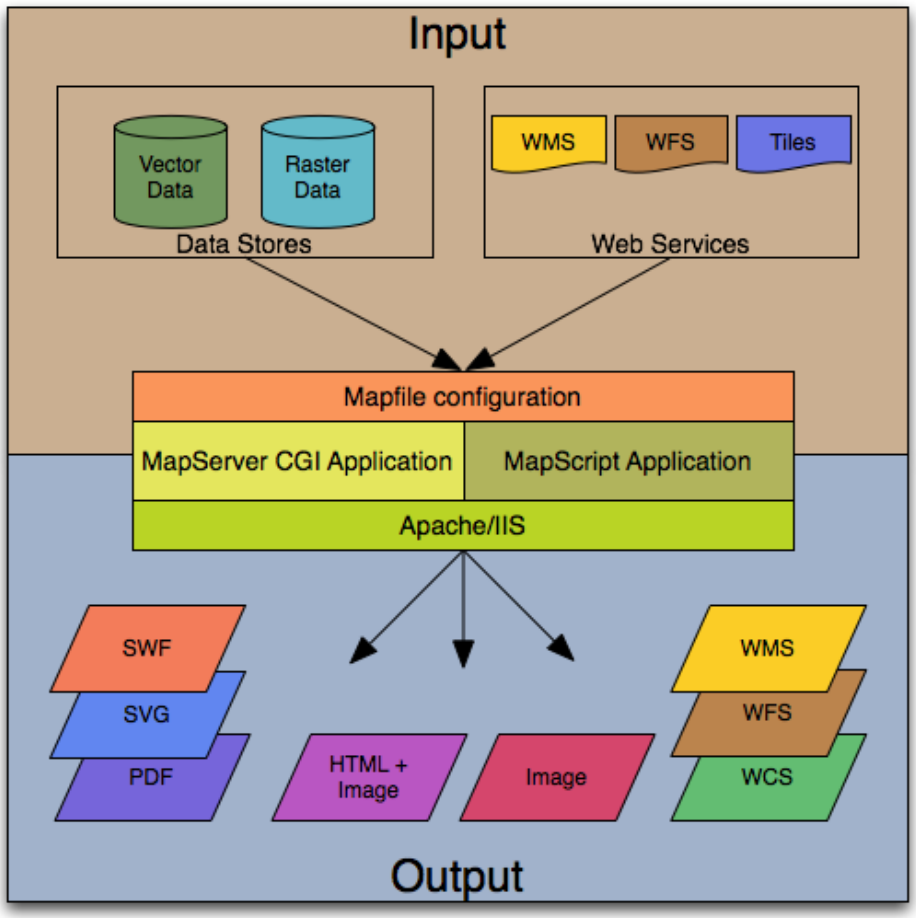


Figura 17: Arquitectura básica de una aplicación de MapServer. Fuente: http://mapserver.org/es/introduction.html

La arquitectura básica consta de una fuente de información, ya sean servicios web o de almacenes de datos, un Mapfile para configurar el mapa en el que se realiza la conexión a los datos, la aplicación CGI a la que el navegador realiza la petición y el servidor web que permite el envío de esa información.

WMS: Web Map Service. Estándar del Open Geospatial Consorium que define como realizar mapas renderizados y devolver al cliente la información. Las peticiones pueden ser hechas desde un navegador web o desde software de escritorio. La información ofrecida es en formato de imagen.

WFS: Web Feature Service. Al igual que WMS es un estándar del Open Geospatial Consortium. En este caso la información es vectorial, incluyendo atributos también.

QuantumGIS 1.7: Software Sistema de Información Geográfica de código abierto que permite trabajar con datos tanto ráster como vectoriales, acceso a diferentes gestores de bases de datos y con la posibilidad de ser configurado con diferentes plugins, uno de ellos para realizar la exportación de mapas a Mapfiles para su representación a través de Mapserver.

Javascript: Lenguaje de programación orientado a objetos que se usa en entorno web.

HTML: siglas en inglés de Lenguaje de Marcado de Hipertexto. Es el lenguaje predominante para elaborar páginas web que se utiliza para describir y traducir la estructura y la información en forma de texto, así como para complementar el texto con objetos tales como imágenes. Puede incluir scripts de Javascript y de PHP.

OpenLayers 2.12: Librería muy completa en lenguaje Javascript que permite acceder y trabajar con información cartográfica. Utilizada para mostrar las capas de MapServer y añadir funcionalidades al visor.

Ext JS 3.4: Librería en Javascript para crear aplicaciones web interactivas.

GeoExt 1.1: Librería en Javascript que aúna OpenLayers 2.12 y Ext JS 3.4 para otorgarle a OpenLayers un entorno más agradable.

Notepad ++: Editor de texto de código abierto con soporte para diferentes lenguajes de programación y opciones avanzadas orientadas a desarrolladores.

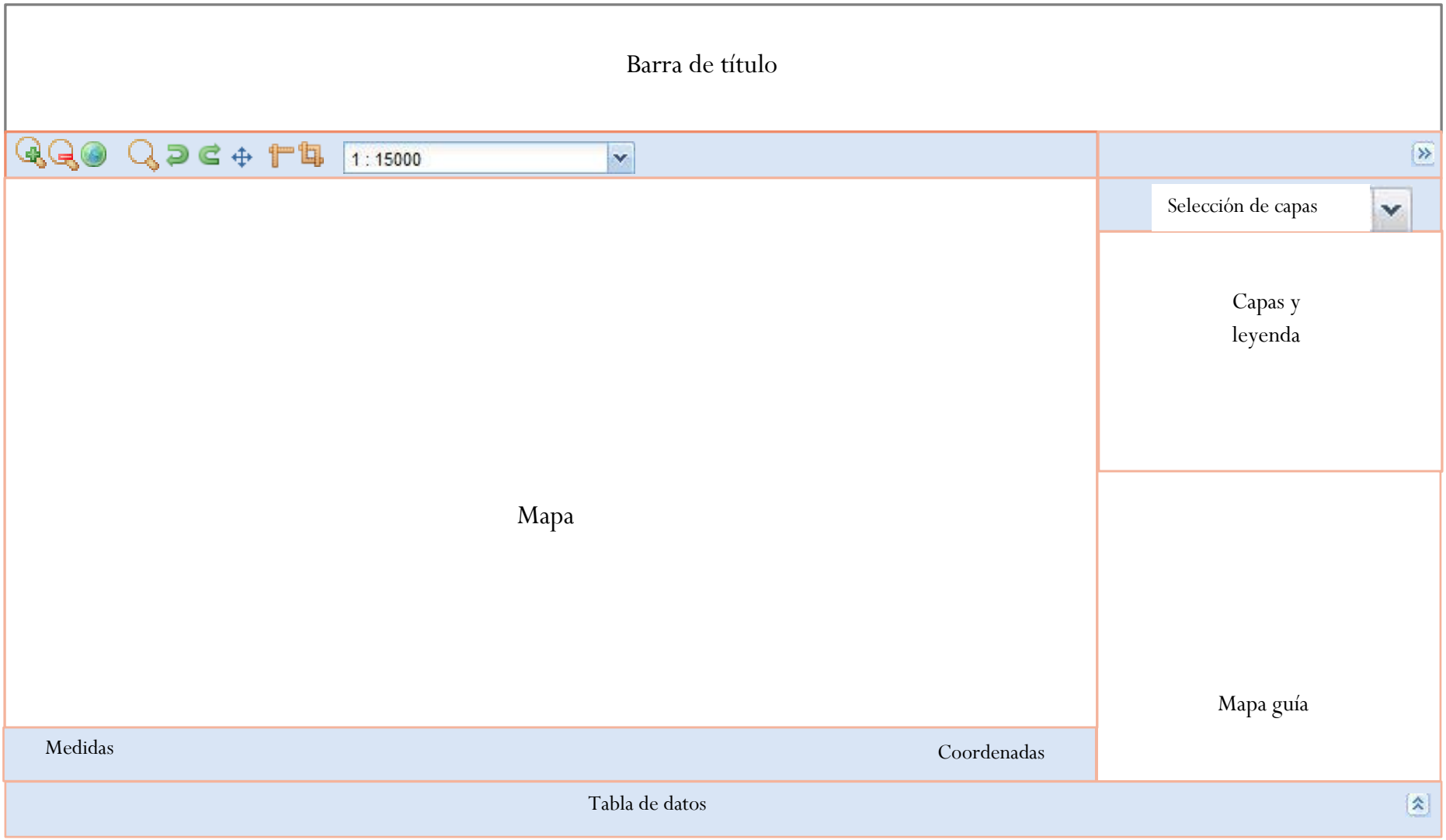
Firefox 17: Navegador web de código abierto.

Firebug 1.11.1: Extensión de Firefox pensada para desarrolladores que permite analizar, editar, monitorizar y depurar el código fuente de una página web.

4.3. Diseño del visor

4.3.1. Diseño de la interfaz

Figura 18



El visor contará con una barra de herramientas sobre el mapa con las funcionalidades de acercar, alejar, zoom a la vista completa, zoom box, vista anterior, vista siguiente, desplazarse, medir líneas, medir áreas y un combo box con varios niveles de escalas. Una barra informativa donde aparecen las coordenadas de donde se encuentra el puntero del ratón y el resultado de las medidas.

También tendrá un panel lateral que se podrá ocultar con un combo box para elegir si se quiere mostrar la capa de tipología o la de conflictividad, un panel con el árbol de capas de dificultades y la capa de municipios, seleccionables mediante checkbox y otro panel con la leyenda de las capas mostradas. En la parte de debajo de este panel lateral habrá otro panel con el mapa guía.

Por último, contará también con una tabla, que de inicio estará oculta, que contendrá información de la base de datos. El usuario podrá elegir que columnas mostrar y cuales ocultar, además podrá ordenar los registros por el atributo que desee.

4.3.2. Información cartográfica

Para cartografiar las comunidades de vecinos se ha usado como capas base las ortofotos del ICC en las escalas 1:25.000, 1:5.000 y 1:2.500, que se muestran según la escala de visualización para verlas a buena resolución sin hacer peticiones demasiado pesadas. También se ha incluido una capa con los municipios que se puede desactivar.

Ortofoto	Escala mínima	Escala máxima
1:25.000	-	15.000
1:5.000	15.000	5.000
1:2.500	5.000	0

Figura 19

En cuanto a la capa de puntos se ha optado por dejar que el usuario elija si quiere ver representadas las comunidades de vecinos por tipología del conflicto o por nivel de conflictividad. Además de eso se puede añadir la representación de cada tipo de las comunidades por cada tipo de conflicto.

También es necesario comentar que la información cartográfica está en el sistema de coordenadas WGS84, que se corresponde con el código EPSG: 4326, ya que las

coordenadas de las comunidades de vecinos se encuentran en este sistema de coordenadas y la cartografía usada como base está disponible en este mismo sistema.

4.4. Visor web

4.4.1. Funcionamiento del visor

El primer paso para crear el portal web es la creación de la cartografía mediante QuantumGIS que permite la exportación a formato .map, el formato propio de MapServer. Se crearán dos archivos .map, uno con las capas que servirán de base y otro con los puntos de las comunidades de vecinos con diferentes simbolizaciones. Además de un archivo OGR Virtual Format, con la extensión .ovf, que permite acceder a la base de datos MySQL y transformar tablas que incluyen coordenadas a información espacial. Posteriormente, mediante las librerías OpenLayers, ExtJS y GeoExt se accede a las capas de MapServer como WMS y WFS para llenar la tabla con los atributos de la capa.

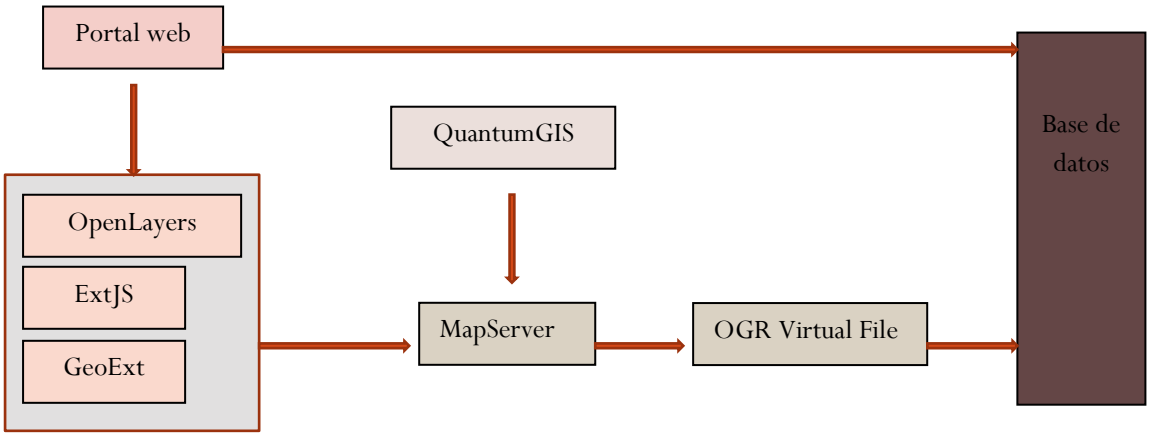


Figura 20

4.4.2. Estructura de archivos y carpetas

La estructura del proyecto cuenta con un archivo index.html y cuatro carpetas en el directorio raíz. En cuanto a las carpetas se han creado cuatro, una con las imágenes, otra con los archivos javascript, una tercera con las librerías y la última con los archivos .map.

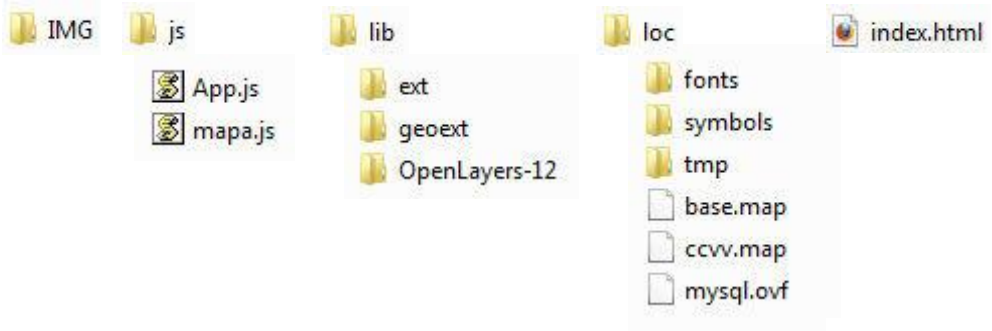


Figura 21

4.4.3. Archivos de MapServer

Como ya se ha explicado, se han realizado dos archivos .map. El primero de ellos contiene las capas base, es decir, las tres ortofotos y la capa de municipios. Las ortofotos se obtienen mediante una conexión WMS al ICC, mientras que la capa de municipios es un archivo shape.

El segundo archivo .map es el que contiene las capas de comunidades de vecinos. En este caso es interesante comentar el acceso a la base de datos para obtener la información de los puntos de comunidades de vecinos. Esta se realiza mediante el driver OGRVRT en lenguaje XML.

El primer paso es acceder a la base de datos y a las tablas que nos interesan, que son tblccvv y tblaccio, en la primera se encuentran las coordenadas mientras que en la segunda los atributos que utilizaremos para representarlas. Se indica que la geometría son puntos y a que columnas corresponden las coordenadas.

```
<OGRVRTDataSource>
 <OGRVRTLayer name="CCVV">
<SrcDataSource>
MYSQL:ccvv,user=root,password=****,host=localhost,port=3306,tables=tblccvv,tblaccio
</SrcDataSource>
<GeometryType>wkbPoint</GeometryType>
<SrcSQL>
SELECT * FROM tblccvv,tblaccio WHERE tblccvv.idCCVV = tblaccio.idCCVV;
</SrcSQL>
<GeometryField encoding="PointFromColumns" x="X" y="Y"/>
</OGRVRTLayer>
</OGRVRTDataSource>
```

Este pequeño archivo ya nos permite simbolizar los puntos con MapServer, en este sentido se crearán diez capas diferentes a partir de los datos extraídos de la base de datos. Las dos primeras capas representarán todas las comunidades de vecinos, una según la tipología del conflicto y la otra según el nivel de conflictividad.

La capa de tipología se representa según el código de tipología de la base de datos, mientras que la de conflictividad indica de forma numérica el nivel de conflictividad del uno al tres.

Las otras ocho capas son las dificultades que se dan en cada comunidad, los campos que representan son binarios por lo que sólo aparecen en la capa las comunidades en la que se da la dificultad seleccionada.

4.4.4. Programación del Visor

4.4.4.1 Organización de archivos

El visor web cuenta con cuatro archivos, además de las librerías correspondientes. El archivo principal es un html que únicamente cuenta con la base de la página y las rutas a las librerías y a los scripts de Javascript. También hay dos archivos Javascript, en el primero se crea el mapa con las capas y las funcionalidades, mientras que en el segundo se programa la aplicación en sí, llamando a las funcionalidades del primer archivo y también añadiendo los paneles, barras y la tabla.

4.4.4.2 Elaboración del código

Ya se ha comentado que hay un archivo en el que define como es el mapa, por él se comenzará la explicación. Este archivo contiene la función llamada crearMapa, que será llamada desde el archivo de la aplicación.

En primer lugar se define la variable mapa como un nuevo OpenLayers.Map y se definen las coordenadas, extensión máxima, escalas, proyección, etc.

```
mapa = new OpenLayers.Map({
    'controls': [],
    div: OpenLayers.Util.getElement('mapa'),
    maxExtent: new OpenLayers.Bounds(2.150000, 41.4300000, 2.250000, 41.4833333),
    restrictedExtent: new OpenLayers.Bounds(2.100000, 41.4000000, 2.300000, 41.5033333),
    scales: [25000,20000,15000,10000,8500,7000,6000,5000,4000,3000,2000,1500,1000,750,500],
    tileSize: new OpenLayers.Size(128,128),
    projection: 'EPSG:4326',
    units: 'degrees'
})
```

Figura 22

A continuación se añaden las capas estarán en el proyecto, se ha puesto como ejemplo la de nivel de conflictividad. Creamos la variable como nueva capa WMS, se le da nombre, se indica la ruta del archivo .map para que se abra desde MapServer y el nombre de la capa. Por último se añade al mapa.

```
conflictivitat = new OpenLayers.Layer.WMS( "Conflictivitat",
"http://localhost/cgi-bin/mapserv.exe?MAP=C:/ms4w/Apache/htdocs/projecte/loc/loCALITZACIO.map&SERVICE=WMS&VERSION=1.1.1",
{layers:'Conflictivitat', transparent:true, format:"image/png"},
{isBaseLayer: false, singleTile:true, visibility: false, displayInLayerSwitcher: true});
mapa.AddLayer(conflictivitat);
```

Figura 23

También se añade la capa de tipología de la intervención, esta capa se añade como capa vectorial y con el protocolo WFS.

```
var styleMap = new OpenLayers.StyleMap({
            'default': {
                graphicOpacity: 1,
                pointRadius: 8
            },
            'select': {
                graphicOpacity: 1,
                pointRadius: 12
            }
});
var lookup = {
        1: {externalGraphic: "IMG/speaker-icon.PNG"},
        2: {externalGraphic: "IMG/basura.PNG"},
        3: {externalGraphic: "IMG/obra.PNG"},
        4: {externalGraphic: "IMG/error.PNG"},
        5: {externalGraphic: "IMG/water-icon.PNG"},
        6: {externalGraphic: "IMG/gestio.PNG"},
        7: {externalGraphic: "IMG/comunicacio.PNG"},
        8: {externalGraphic: "IMG/ascensor.gif"},
        9: {externalGraphic: "IMG/aire.PNG"}
    };
    styleMap.addUniqueValueRules ("default", "IdTipMed", lookup);

tipologia = new OpenLayers.Layer.Vector( "Tipologia",{
    strategies: [new OpenLayers.Strategy.BBOX()],
    extractAttributes:true,
    styleMap: styleMap,
    protocol: new OpenLayers.Protocol.WFS({
        url:"http://localhost/cgi-bin/mapserv.exe?MAP=C:/ms4w/Apache/htdocs/projecte/loc/loCALITZACIO.map",
        featureType: "Tipologia",
        featureNS: "http://mapserver.gis.umn.edu/mapserver"
    }),
    displayInLayerSwitcher: false
});
```

Figura 24

En la imagen se aprecia como primero se declara el objeto de estilo, diferenciando como se ve por defecto y cuando está seleccionado, que se ve más grande. Posteriormente hay una variable que indica el icono para cada valor del campo IdTipMed, y se añade la regla de valores únicos al estilo. Por último se define la capa vectorial.

Lo siguiente son los controles de OpenLayers. Se añadirán el control de Acercar zoom, alejar zoom, la navegación por el mapa, el zoom a la máxima extensión, zoom al rectángulo, el historial de vistas y las herramientas de medición de línea y de área. A continuación se puede apreciar el ejemplo de uno de ellos, en el que primero se crea la variable y después se añade el control al mapa.

```
ctrZoomOut = new OpenLayers.Control.ZoomOut();
mapa.addControl(ctrZoomOut)
```

Figura 25

En el otro archivo de Javascript se crea el visor web. En primer lugar se realiza la función onReady que lo primero que hace es ejecutar la función crearMapa que contiene todos los parámetros relativos al mapa, así como las capas de información. Se añade la leyenda con un panel de GeoExt en el cual se indica de que mapa es la leyenda, el título del panel, el estilo, un filtro para que solo aparezcan las capas seleccionadas en la leyenda, donde debe ir dentro del viewport, tamaño y otras propiedades de la leyenda.

```
legendPanel = new GeoExt.LegendPanel({
    map: mapa,
    defaults: {
        labelCls: 'mylabel',
        style: 'padding:5px',
        baseParams:{
            FORMAT:'image/png',
            LEGEND_OPTIONS:'forceLabels:on'
        }
    },
    filter:function(record){
        return record.getLayer().displayInLayerSwitcher;
    },
    useArrows: true,
    animate:true,
    enableDD:true,
    collapsible:true,
    expanded:false,
    bodyStyle: 'padding:5px',
    width: 350,
    autoScroll: true,
    region: 'west',
    title: 'Llegenda',
    dynamic: true
});
```

Figura 26

A continuación se añade el mapa guía, con el tamaño en píxeles y las coordenadas, el comboBox para cambiar el zoom que permitirá elegir una escala determinada.

Después de esto se crea el objeto Viewport de la librería Ext. Este objeto se renderiza en el cuerpo del documento y se encarga de organizar el espacio del visor dentro de la ventana del navegador ajustándose automáticamente a ésta. Todos los paneles de Ext tendrán definida su posición dentro del Viewport.

En la definición de la variable App, que es en la que se crea el Viewport, se crea el espacio para el título llamando a la función crearTitulo, se llama también a la función que crea el MapPanel de GeoExt, se crea el panel que contiene el combobox para elegir entre las capas de tipología y nivel de conflictividad, el árbol de capas, la leyenda y el mapa guía y por último la tabla.

El siguiente paso es hacer el gridPanel que incluirá la tabla de MySQL, para ello utilizaremos los atributos de la capa WFS. Para ello se usará un gridPanel de Ext con la capa tipología como almacén de datos al que le debemos indicar los campos de la tabla, especificando el nombre y el tipo de cada campo. Dentro del objeto de la tabla también debe ser definida cada columna relacionándola con las columnas del almacén de datos de la capa. En algunas de las columnas se añade la propiedad ocultarlas y que el usuario las pueda seleccionar para verlas, así se evita un exceso de información al ver la tabla. También se permite que el usuario ordene los registros de la tabla por el campo que prefiera. Al añadir los registros de la tabla desde la capa WFS permite seleccionar un punto y que se seleccione automáticamente en la tabla y viceversa.

```
grid = new Ext.grid.GridPanel({
    sm: new GeoExt.grid.FeatureSelectionModel(),
    store: new GeoExt.data.FeatureStore({
        layer: tipologia,
        fields: [
            {name:'IdCCVV', type: 'int'},
            {name:'TipusVia', type: 'string'},
            {name:'NomVia', type: 'string'},
            {name:'Num1', type:'int'},
            {name:'Num2',type:'int'},
            {name:'IdAccio',type:'int'},
            {name:'IdNivelConf',type:'int'},
            {name:'NumDomicili',type:'int'},
            {name:'NumPersones',type:'int'},
            {name:'NumComer',type:'int'},
            {name:'DifConvivencia',type:'string'},
            {name:'DifSorolls',type:'string'},
            {name:'DifOrganitzacio',type:'string'},
            {name:'DifMalusEspaisComuns',type:'string'},
            {name:'DifImpagaments',type:'string'},
            {name:'DifNeteja',type:'string'},
            {name:'DifSobreocupacio',type:'string'},
            {name:'DifAltres',type:'string'},
            {name:'TipMed',type:'string'}
        ],
        proxy: new GeoExt.data.ProtocolProxy({
            protocol: new OpenLayers.Protocol.WFS({
                url: "http://localhost/cgi-bin/mapserv.exe?MAP=C:/ms4w/Apache/htdocs/projecte/loc/loCALITZACIO.map",
                version: "1.0.0",
                FeatureType: "Tipologia",
                FeatureNS: "http://mapserver.gis.umn.edu/mapserver",
                srsName: "EPSG:4326"
            })
        }),
```

Figura 27

Ya se ha hablado de un combobox para seleccionar que simbolización de las comunidades de vecinos mostrar, si la de tipología del conflicto o la de nivel de conflictividad. Esto se ha realizado con el código que se puede ver en la figura 28, en él se aprecia un almacén con los campos de valor y nombre de la capa. A continuación se programa la función onSelect con un if, que hará que cuando se seleccione una capa se oculte la otra y se muestre esa.

```
var cbTipus=new Ext.form.ComboBox({
        name:'cbTipus',
        hiddenName:'cbTipus',
        store:new Ext.data.SimpleStore({
                fields: ['value','tipus'],
                data:[
                        [0,'Tipologia'],
                        [1,'Nivell de conflictivitat']
                ]
        }),
        mode: 'local',
        fieldLabel:'Fons',
        forceSelection: true,
        value:0,
        valueField:'value',
        displayField:'tipus',
        editable:false,
        triggerAction:'all'
});
    cbTipus.on('select',
        function(combo,record,index){
            if(record.get('value') == 0){
                mapa.layers[3].setVisibility (false);
                mapa.layers[2].setVisibility (true);
            }else{
                mapa.layers[2].setVisibility (false);
                mapa.layers[3].setVisibility (true);
            }
        },
        this
    );
```

Figura 28

Ahora se realizará la función que crea todo el panel del mapa, que incluye, además del mapa en sí, todas las funcionalidades para interactuar con éste.

Lo primero que se realizará es configurar las herramientas de visualización creadas en el archivo Javascript del mapa, en la figura 29 se ve como se crea una variable como

Acción de GeoExt, añadiendo el control de OpenLayers, el icono y la etiqueta que aparecerá cuando el cursor esté sobre el botón.

```
var actZoomOut = new GeoExt.Action({
  control: ctrZoomOut,
  icon:"IMG/zoom_out.ico",
  tooltip: "Alejar"
  });
```

Figura 29

En la figura 30 está el código que crea el panel del mapa, se llama al objeto mapa, se sitúa las coordenadas del centro del mapa, el nivel de zoom por defecto, en qué región del Viewport estará situado y una barra de escala.

```
mapPanel = new GeoExt.MapPanel({
  map: mapa,
  center: new OpenLayers.LonLat(2.211567, 41.450851),
  zoom: 1,
  region: 'center',
  items: [{
    xtype: "gx_zoomslider",
    vertical: true,
    minValue: 0,
    maxValue: 14,
    height: 140,
    x: 10,
    y: 20,
    plugins: new GeoExt.ZoomSliderTip({template:"1:{scale}"})
  }],
```

Figura 30

Además de las características del mapa se añaden la barra de herramientas superior con todos los botones y la inferior con las coordenadas y las medidas. Con esto ya estaría creado el panel del mapa.

También se debe realizar el título, en la región norte del Viewport y con el contenido *titulo* que es un div de html y que incluye las imágenes del título.

```
function crearTitulo() {
  return new Ext.BoxComponent({
    region: 'north',
    contentEl: 'titulo',
    height: 65
  })
}
```

Figura 31

Ahora se creará el árbol de capas, desde donde se podrán activar y desactivar las capas de tipologías de dificultades de las comunidades de vecinos y la de municipios. Para realizar el árbol de capas en primer lugar se crea una variable con la clase LayerNodeUI de Ext y se extiende con las subclases de GeoExt LayerNodeUI y TreeNodeUIEventMixin que añade eventos, así una capa cuando sea activada se mostrará al momento.

```
var LayerNodeUI = Ext.extend(GeoExt.tree.LayerNodeUI, new GeoExt.tree.TreeNodeUIEventMixin());
```

Figura 32

También han sido creadas dos variables para configurar el árbol de capas, una llamada ccvv, con las capas de las comunidades de vecinos y otra que solo contendrá la capa de límites municipales. La figura 33 es un ejemplo de la configuración del árbol de capas, en ella se define la raíz y una de las capas, en concreto la de *Mal ús de l'espai comú*, en ella se dice que el nodo es una capa de GeoExt, el texto que se verá, que no estará activa por defecto y que capa queremos que se muestre.

```
var ccvv ={
        text: 'Comunitats de veïns',
        cls: 'folder',
        expanded: true,
        children: [{
            nodeType: "gx_layer",
            text: "Mal ús de l'espai comú",
            checked: false,
            layer: "Mal ús de l'espai comú",
            leaf: true,
            loader: {param: "LAYERS"}
            },
```

Figura 33

A continuación se convierte esta configuración a formato JSON y se guarda en una variable nueva.

```
var treeConfig = new OpenLayers.Format.JSON().write([base,ccvv], true)
```

Figura 34

Por último se realiza la función que crea el árbol de capas (Figura 35), que devuelve un nuevo Tree Panel. Se indica la región del Viewport, nombre, tamaño, etc y como provider la variable LayerNodeUI, que se trata de esta misma clase de Ext modificada, como ya se ha explicado. Mientras que en la propiedad root se define la raíz de las

capas, que como ya han sido antes convertidas a JSON se llama al objeto Decode de Ext y se le pasa la variable donde está guardado el JSON.

```
function crearArbolCapas() {
  return new Ext.tree.TreePanel({
    region: "east",
    title: "Capas",
    width: 250,
    autoScroll: true,
    lines: false,
    loader: new Ext.tree.TreeLoader({
        applyLoader: false,
        uiProviders: {
            "layernodeui": LayerNodeUI
        }
    }),
    root: {
      nodeType: "async",
      children: Ext.decode(treeConfig)
    },
    rootVisible: false
  })
};
```

Figura 35

Por último se explicará cómo se muestran las medidas en la barra de herramientas. Se crea una función y dentro de ella la variable txt que le da nombre al objeto para poder ser llamado mostrado en la barra de herramientas. Y una condicional que indica que si el orden es igual a uno aparecerá el texto *Distancia:* y la distancia medida, y si no aparecerá *Área:* las unidades y ^{2} para indicar que es el área.

```
function displayMeasure(evt) {
  var txt = Ext.getCmp('medTxt');
  if (evt.order==1) {
    txt.update("Distancia: "+ Math.round(evt.measure*100)/100 + " " + evt.units);
  } else {
    txt.update("Área: "+ Math.round(evt.measure*100)/100 + " " + evt.units + "<sup>2</sup>");
  }
}
```

Figura 36

4.5. Resultados

Figura 37: Vista general del visor cartográfico

En la figura 37 se aprecia como es el visor una vez acabado, ahora se explicarán las funcionalidades implementadas.

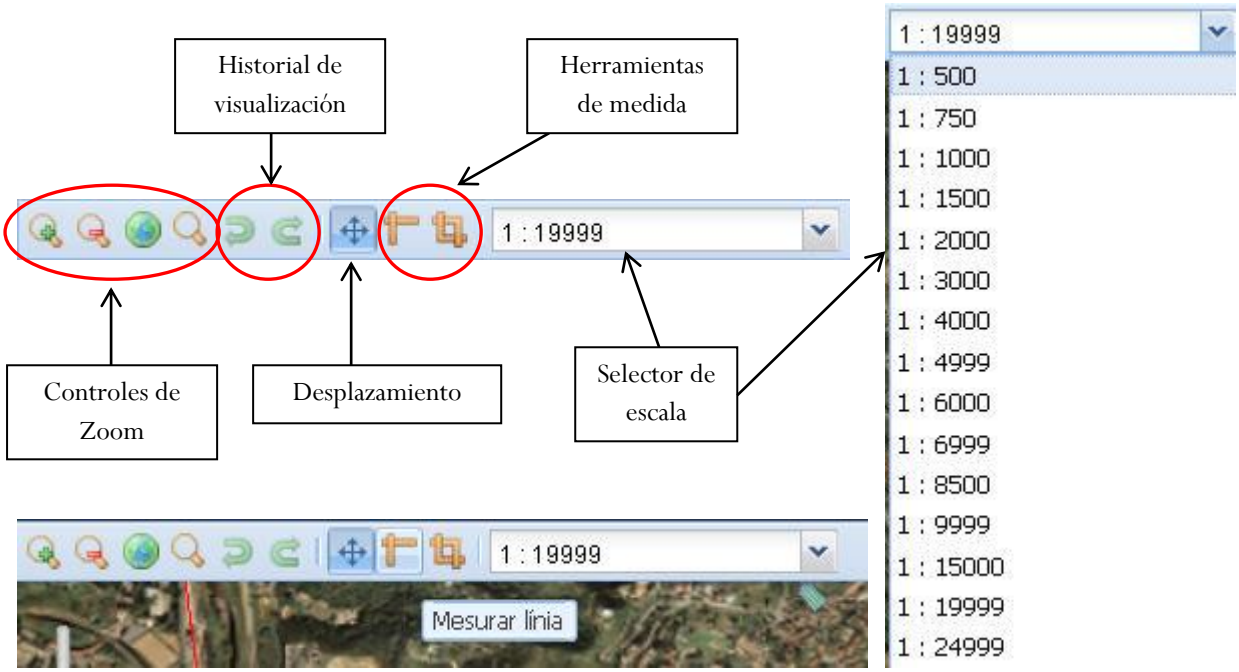


Figura 38: Barra de herramientas explicada (arriba), mostrando una etiqueta (abajo) y el selector de escala extendido (derecha)

En la figura 38 se ve la barra de herramientas, con cada botón explicado, una muestra de las etiquetas o Tool tip y el selector de escala extendido.

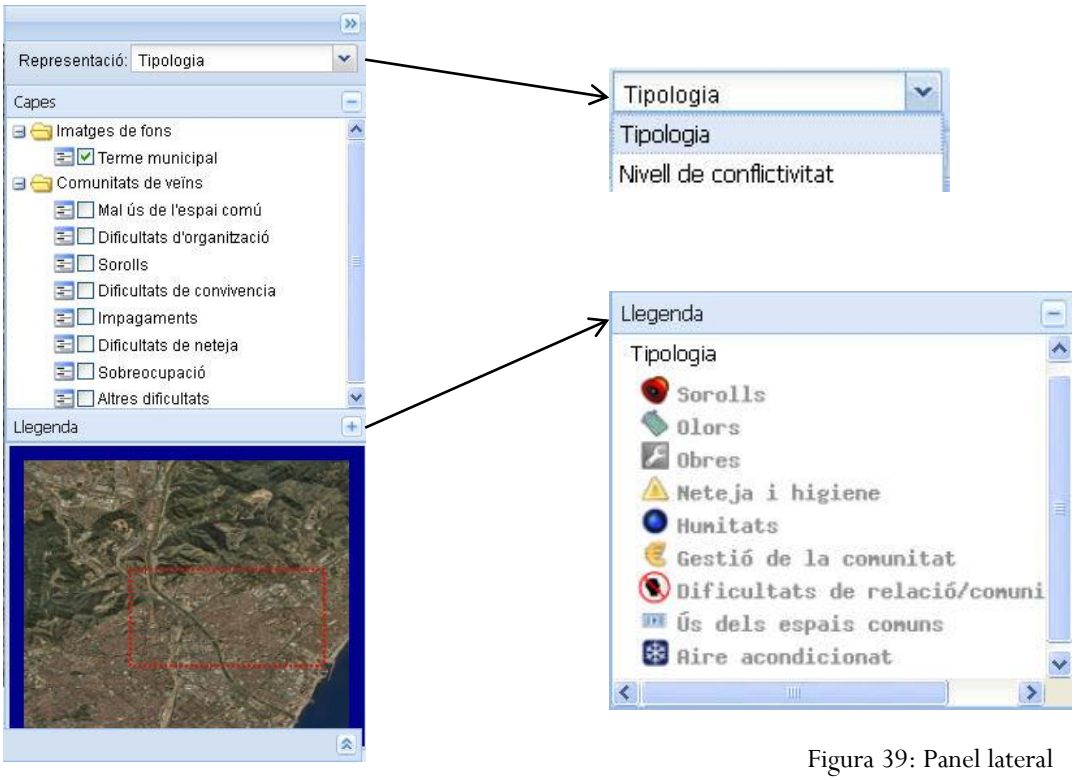


Figura 39: Panel lateral

En la figura 39 se explica el panel lateral con el selector de capas desplegado arriba a la derecha y la leyenda abajo a la derecha.

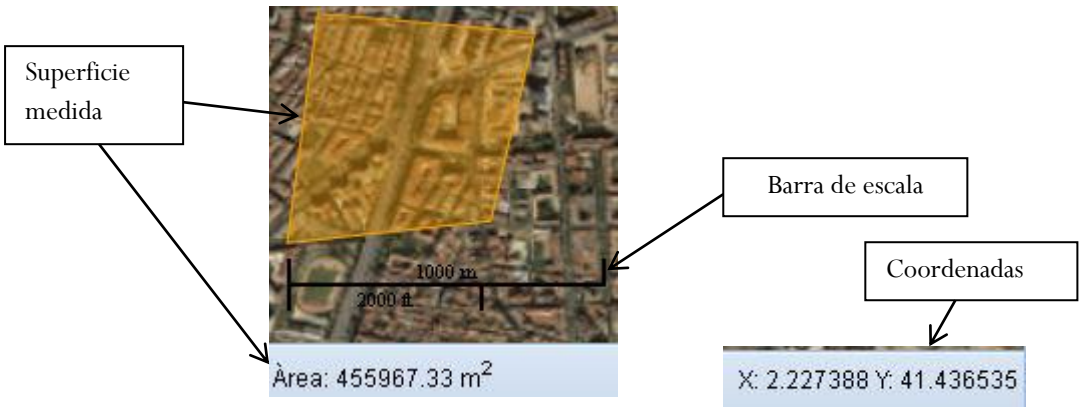


Figura 40: Barra inferior con medidas y coordenadas

En la figura 40 se muestran los elementos de la barra inferior, el texto del área sólo aparece cuando el usuario ha creado un polígono o una línea, aunque en este último caso no aparece el texto área, pero sí la medida en metros. También aparece la barra de escala que está dentro del objeto mapa y por último las coordenadas.

El último de los elementos es la tabla de atributos de las comunidades con los campos que aparecen por defecto y las columnas que pueden ser mostradas,

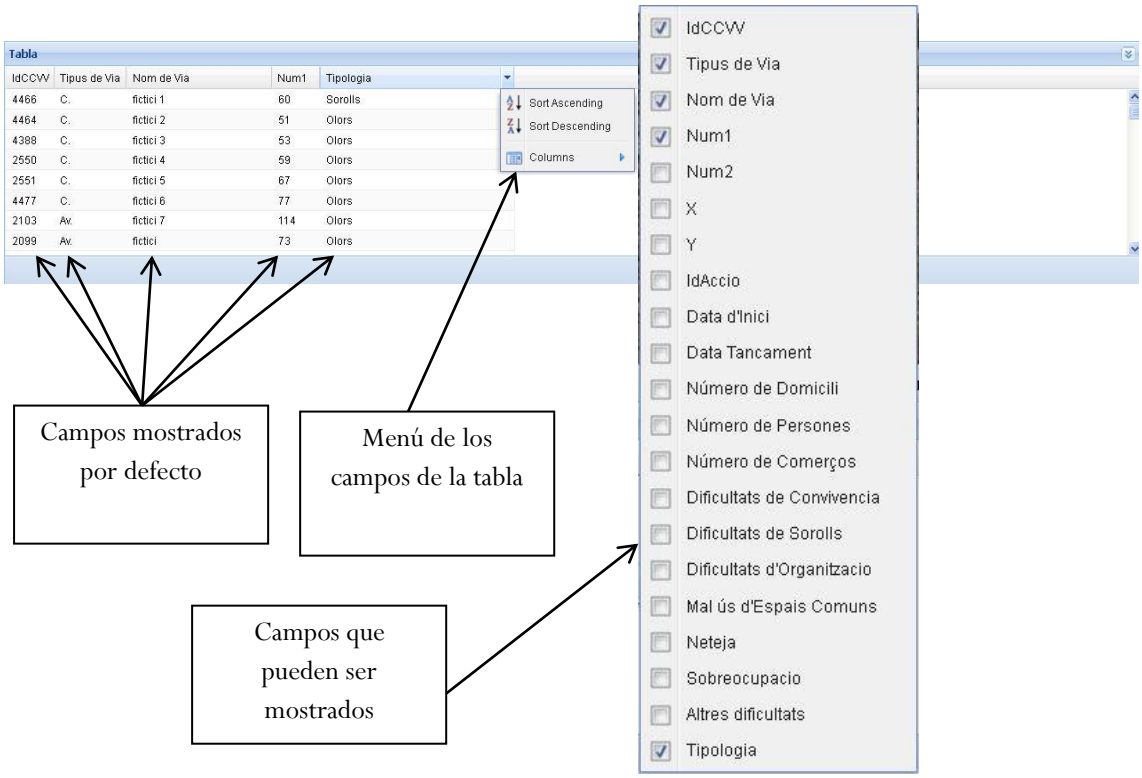


Figura 41: Tabla de atributos

A continuación se explica la selección de elementos. En la figura 42 se muestra una parte del mapa y la tabla de atributos de la capa WFS, sin seleccionar ningún elemento. En la figura 43 sí que hay un registro seleccionado y se aprecia como su icono es más grande que el resto.

Tabla

Tipologia	IdCCVV	Nivell de c...	Tipus de Via	Nom de Via	Num1
Neteja i higiene	4458	2	C.	fictici	39
Neteja i higiene	3947	2	Ptge.	fictici	5
Neteja i higiene	4375	2	C.	fictici	14
Neteja i higiene	4374	2	C.	fictici	12
Neteja i higiene	3963	3	C.	fictici	11
Humitats	3967	3	C.	fictici	24
Humitats	3969	3	C.	fictici	28
Humitats	2358	3	C.	fictici	54

Figura 42: Tabla de atributos y el mapa sin selección

Tabla

Tipologia	IdCCVV	Nivell de c...	Tipus de Via	Nom de Via	Num1
Neteja i higiene	4458	2	C.	fictici	39
Neteja i higiene	3947	2	Ptge.	fictici	5
Neteja i higiene	4375	2	C.	fictici	14
Neteja i higiene	4374	2	C.	fictici	12
Neteja i higiene	3963	3	C.	fictici	11
Humitats	3967	3	C.	fictici	24
Humitats	3969	3	C.	fictici	28
Humitats	2358	3	C.	fictici	54

Figura 43: Tabla de atributos y mapa con registro seleccionado

La tabla permite selección múltiple, únicamente se debe presionar la tecla ctrl cuando se vaya a seleccionar. Si no se presiona el registro seleccionado anteriormente se deselecciona.

La selección de registros sólo se permite con la capa de tipología, si la que se muestra es la capa de conflictividad los elementos no pueden ser seleccionados.

Conclusiones

Los objetivos que planteaba este proyecto en un principio han sido alcanzados en un alto grado, aunque se podrían añadir nuevas funcionalidades al visor como la posibilidad de añadir información a la base de datos desde el mismo. La aplicación de actualización de datos, el objetivo principal era que actualizara los datos de una forma rápida sin interferir en el rendimiento del servidor, algo que se ha conseguido, pues el tiempo de actualización para la muestra es mucho menor de un segundo, aunque la muestra sean únicamente dos tablas no hay motivos para pensar que para actualizar la base de datos al completo pueda interferir en el rendimiento del servidor, estando, además, programada para ejecutarse a las 0 horas, que el servidor no está en uso.

La tecnología utilizada ha sido la adecuada, aunque el desarrollo del proyecto no ha estado exento de dificultades tanto con MapServer como con OpenLayers, dificultades que finalmente fueron solventadas gracias a la documentación aportada por las páginas web de ambos proyectos. Respecto a estos problemas se debe reconocer que quizá habría sido más adecuado utilizar Geoserver en lugar de MapServer, ya que cuenta con una interfaz mucho más amigable. En cuanto al uso de Visual Studio 2010 para realizar la aplicación de actualización de datos, no han surgido dificultades más allá del desconocimiento del estudiante del conector de MySQL, aunque fueron fácilmente solucionados gracias a la documentación existente.

La interfaz del visor cartográfico es muy amigable y sólo requiere de unos conocimientos básicos de navegación por internet para su uso. Los tooltips que hay en los botones también sirven de ayuda para saber para qué sirven mientras que la leyenda aclara el significado de cada símbolo.

Quizá sería posible una mejora del visor para agrupar los puntos cercanos y que con un click se desagrupe y permita ver todos los puntos de esa zona. Además de esto, también se podrían crear filtros para la tabla y mostrar sólo los registros que necesite visualizar el usuario.

También se puede plantear una posibilidad para el futuro, que sería realizar una aplicación móvil que facilite la introducción de puntos con la información necesaria y el abandono definitivo de la base de datos de Microsoft Acces para usar únicamente la de MySQL que tiene mucho más potencial.

Referencias bibliográficas

Documentación del conector de MySQL para Microsoft Visual Basic 2010 Express. http://dev.mysql.com/doc/refman/5.5/en/connector-net-visual-studio.html

Documentación de MySQL.
http://dev.mysql.com/doc/refman/5.5/en/index.html

Documentación de phpMyAdmin.
https://phpmyadmin-spanish.readthedocs.org/en/latest/

Documentación del driver OGR Virtual Format.
http://www.gdal.org/ogr/drv_vrt.html

Documentación de MapServer.
http://mapserver.org/MapServer-56.pdf

Documentación de OpenLayers.
http://dev.openlayers.org/releases/OpenLayers-2.12/doc/apidocs/files/OpenLayers-js.html

Documentación ExtJS 3.4.
http://docs.sencha.com/ext-js/3-4/#!/api

Documentación GeoExt.
http://www.geoext.org/lib/index.html